IPW

PTO/SB/21 (08-00)
Approved for use through 10/31/02. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMENRCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

		Application Number	10/670,901
		Filing Date	September 24, 2003
		First Named Inventor	Philippe Jung
		Group Art Unit	2183
		Examiner Name	NYA
Total Number of Pages in This Submission	34	Attorney Docket Number	15437-0639

ENCLOSURES ***(check all that apply)***

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment / Response
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☒ Certified Copy of Priority Document(s)

  from France dtd 13 Juin 2003
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Assignment Papers *(for an Application)*
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition To Convert To a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, number of CD(s)_____
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☐ Other Enclosure(s) *(please identify below):*

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	Hickman Palermo Truong & Becker LLP Craig G. Holmes
Signature	
Date	June 3, 2004

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class: mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this date: 6/3/04

Type or printed name	Annette Jacobs		
Signature		Date	June 3, 2004

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

P8336

R E P U B L I Q U E F R A N Ç A I S E

02 11076
Lettres (1)

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 13 JUIN 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr

DB 267/141102
ETABLISSEMENT PUBLIC NATIONAL CREE PAR LA LOI N° 51-444 DU 19 AVRIL 1951

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE
26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

cerfa N° 11354*01

Code de la propriété intellectuelle - Livre VI

REQUÊTE EN DÉLIVRANCE 1/2

Important ! Remplir impérativement la 2ème page.

Cet imprimé est à remplir lisiblement à l'encre noire 08 540 W / 190600

Réservé à l'INPI

REMISE DES PIÈCES
DATE 30 SEPT 2002
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0212076

DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 30 SEP. 2002

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

CABINET NETTER
36 avenue Hoche
75008 PARIS

Vos références pour ce dossier
(facultatif) SUN 46 (120772)

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	**Cochez l'une des 4 cases suivantes**
Demande de brevet	☒
Demande de certificat d'utilité	☐
Demande divisionnaire	☐
Demande de brevet initiale	N° Date / /
ou demande de certificat d'utilité initiale	N° Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N° Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Improved filtering of redundant packets in computer network equipments.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE

Pays ou organisation
Date / / N°

Pays ou organisation
Date / / N°

Pays ou organisation
Date / / N°

☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**

5 DEMANDEUR ☐ **S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»**

Nom ou dénomination sociale		SUN MICROSYSTEMS, INC	
Prénoms			
Forme juridique			
N° SIREN			
Code APE-NAF			
Adresse	Rue	901 San Antonio Road	
	Code postal et ville	94303	PALO ALTO Californie
Pays		Etats-Unis d'Amérique	
Nationalité		américaine	
N° de téléphone *(facultatif)*			
N° de télécopie *(facultatif)*			
Adresse électronique *(facultatif)*			

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DES PIÈCES
DATE 30 SEPT 2002
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT
NATIONAL ATTRIBUÉ PAR L'INPI 0212076

DB 540 W /190600

Vos références pour ce dossier : *(facultatif)*		SUN 46 (120772)
6 MANDATAIRE		
Nom		PLAÇAIS
Prénom		Jean-Yves
Cabinet ou Société		Cabinet NETTER
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	36 avenue Hoche
	Code postal et ville	75008 PARIS
N° de téléphone *(facultatif)*		01 58 36 44 22
N° de télécopie *(facultatif)*		01 42 25 00 45
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		
Les inventeurs sont les demandeurs		☐ Oui ☒ Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☐ ☒
Paiement échelonné de la redevance		**Paiement en deux versements, uniquement pour les personnes physiques** ☐ Oui ☐ Non
RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence) :*
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE
(Nom et qualité du signataire)

N° Conseil 92-1197 (B) (M)
Jean-Yves PLAÇAIS

VISA DE LA PRÉFECTURE OU DE L'INPI

M. BLANCANEAUX

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire.
Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

sun46.FRD.wpd

P8336

Improved filtering of redundant packets in computer network equipments

This invention relates to network equipment, as used for example in telecommunication systems.

In such equipment, computer workstations or nodes are interconnected through a network medium or link. The link may have to be at least partially duplicated to meet reliability constraints. This duplication is called link redundancy. It is now assumed by way of example that data are exchanged between the nodes in the form of packets. Considering a given packet sent from a source node to a destination node, redundancy means that two or more copies of that packet are sent to the destination node through two or more different networks, respectively. The copies of the packet will usually reach the destination node at different times. Thus, the first of the packets is processed normally in the destination node; when arriving, the other copy or copies ("redundant packets") are processed in a manner which may depend on the transport protocol and/or the user application.

The Transmission Control Protocol (TCP) has a built-in capability to suppress redundant packets. However, this built-in capability involves potentially long and unpredictable delays. On another hand, the User Datagram Protocol (UDP) has no such capability; in this case, suppressing redundant packets is a task for user applications.

A general aim of the present invention is to provide advances with respect to such mechanisms.

In a first aspect, this invention offers software code, comprising code for defining:

- a memory manager, having a reserved memory area, in which it defines a database comprising a set of portions of database, each portion being designated with a first index value, and
- an incoming packet manager, responsive to an incoming packet for:
    * determining an identifier from data contained in the incoming packet and a first value from said identifier,

α

* searching the identifier in a portion of database designated with a first index value corresponding to the first value, said portion being called the found portion of database :

* if the identifier is not found in said first portion of database, storing the identifier in said found portion of database,

* else, if a time condition is met, removing the identifier from said found portion of database, the incoming packet being a redundant packet.

In a second aspect, this invention offers a method of processing redundant packets comprising the steps of :

a- reserving memory area for a database comprising a set of portions of database, each portion being designated with a first index value,

b- determining an identifier from data contained in the incoming packet and a first value from said identifier,

c- searching the identifier in a portion of database designated with a first index value corresponding to this first value, said portion of database being called the found portion of database :

c1- if the identifier is not found in said first portion of database, storing the identifier in said found portion of database,

c2- else, if a time condition is met, removing the identifier from said found portion of database, the incoming packet being a redundant packet.

Other alternative features and advantages of the invention will appear in the detailed description below and in the appended drawings, in which :

- figure 1 is a general diagram of a telecommunication network system, as an exemplary context in which this invention may be applicable;

- figure 2 shows a group of stations or nodes interconnected through two different links;

- figure 3 is a block diagram of a computer station or node, incorporating the invention;

- figure 4a shows a first exemplary format of an IPv4 header in a packet;

- figure 4b shows a second exemplary format of an IPv6 header in a packet;

- figure 5 shows a flow chart of a packet transmission in redundant mode;

- figure 6 shows the initial reception of redundant packets;

- figure 7 shows the structure of an exemplary software module used in this invention;

- figure 8 is a general flow-chart of the discrimination of received packets;

- figure 9 shows an exemplary of source node table;

- figure 10 is an exemplary of database according to the invention.

As they may be cited in this specification, Sun, Sun Microsystems, Solaris, ChorusΘS are trademarks of Sun Microsystems, Inc. SPARC is a trademark of SPARC International, Inc.

Additionally, the detailed description is supplemented with the following Exhibits:

- Exhibit A contains code examples used in this specification.

In the foregoing description, references to the exhibit and to the drawings nevertheless form an integral part of the description of the present invention.

Making reference to software entities imposes certain conventions in notation. For example, in the detailed description, *Italics* (and/or the quote sign ") may be used when deemed necessary for clarity to designate specific functions.

Figure 1 illustrates an exemplary simplified telecommunication network system. Terminal devices (TD) like 1 are in charge of transmitting data, e.g. connection request data, to base transmission stations (BTS) like 3. A such base transmission station 3 gives access to a communication network, under control of a base station controller (BSC) 4. The base station controller 4 comprises communication nodes, supporting communication services ("applications"). Base station controller 4 also uses a mobile switching center 8 (MSC), adapted to orientate data to a desired communication service (or node), and further service nodes 9 (General Packet Radio Service, GPRS), giving access to network services, e.g. Web servers 19, application servers 29, data base server 39. Base station controller 4 is managed by an operation management center 6 (OMC).

Certain items in the system of figure 1, e.g. the base station controllers 4, may comprise one or more groups of nodes, or clusters, exchanging data through two or more redundant networks. Reference to base station controllers is purely exemplary, since other components in a telecommunication system or systems, as well as in other types of data or message exchanging system or systems, may have a similar organization.

Figure 2 shows a cluster having D nodes N1, N2, .. N_D, interconnected through two different links 31 and 32. In the foregoing description, Ni and Nj designate two nodes, with i and j being comprised between 1 and D, inclusively. The network links or channels 31, 32 as used may be high speed network channels with equivalent bandwidth and latency. However, other channels may be also used, e.g. heterogeneous networks. As example only, links 31 and 32 are arranged as Ethernet physical networks. Other links may be possible such over ATM (Asynchronous Transfer Mode) or faster links such as InfiniBand.

Figure 3 shows an exemplary node Ni, in which the invention may be applied. Node Ni comprises, from top to bottom, applications 13, management layer 11, network protocol stack 10, and link level interfaces 12 and 14, respectively interacting with network links 31 and 32 (also shown in figure 2). Node Ni may be part of a local or global network; in the foregoing exemplary description, the network is the Internet, by way of example only. It is assumed that each node may be uniquely defined by a portion of its Internet address. Accordingly, as used hereinafter, “Internet address” or “IP address” means an address uniquely designating a node in the network being considered (e.g. a cluster), whichever

network protocol is being used. Although the Internet is convenient at present, there is no restriction to the Internet.

Thus, in the example, network protocol stack 10 comprises:

- an Internet interface 100, having conventional Internet protocol (IP) functions 102, and a multiple data link interface 101,
- above Internet interface 100, message protocol processing functions, e.g. an UDP function 104 and/or a TCP function 106.

Network protocol stack 10 is interconnected with the physical networks through first and second link level interfaces 12 and 14, respectively. These are in turn connected to first and second network channels 31 and 32, via couplings L1 and L2, respectively. More than two channels may be provided, enabling to work on more than two copies of a packet.

Link level interface 12 has an Internet address *<IP_12>* and a Link level address *<<LL_12>>*. Incidentally, the doubled triangular brackets (<< ... >>) are used only to distinguish link level addresses from Internet addresses. Similarly, Link level interface 14 has an an Internet address *<IP_14>* and a Link level address *<<LL_14>>*. In a specific realization, where the physical network is Ethernet-based, interfaces 12 and 14 are Ethernet interfaces, and *<<LL_12>>* and *<<LL_14>>* are Ethernet addresses.

IP functions 102 comprise encapsulating a message coming from upper layers 104 or 106 into a suitable IP packet format, and, conversely, de-encapsulating a received packet before delivering the message it contains to upper layer 104 or 106.

In redundant operation, the interconnection between IP layer 102 and link level interfaces 12 and 14 occurs through multiple data link interface 101. The multiple data link interface 101 also has an IP address *<IP_10>*, which is the node address in a packet sent from source node Ni.

References to Internet and Ethernet are for example only, and other protocols may also be used, both in stack 10, including multiple data link interface 101, and/or in link-level interfaces 12 and 14.

α

Furthermore, where no redundancy is required, IP layer 102 may directly exchange messages with anyone of interfaces 12,14, thus by-passing multiple data link interface 101.

When circulating on any of links 31 and 32, a packet may have several layers of headers in its frame: for example, a packet may have, encapsulated within each other, a transport protocol header, an IP header, and a link-level header.

As shown in figure 4a, a first exemplary header (IPv4) may comprise the following fields:
- a destination IP address 220;
- a source IP address 221;
- a header checksum 222;
- a Time To Live (TTL)223;
- a protocol identifier (*IP-PROT*) 224;
- a zone 225 containing fragmentation flags, and fragment offsets (*IP-OFF*);
- an IP identification (*IP-ID*) 226;
- an IP total length 227;
- a type of service (T.O.S.) 228;
- a Header Length (Internet Header Length) 229; and
- a version identifier 230 of a protocol being in this case the internet protocol version 4 (IPv4).

Certain of these fields are defined at the level of network protocol stack 10. For a packet corresponding to a complete data message, fields 220, 221, 224 and 226 are sufficient to identify the data message. Optionally, a data message may be split into a plurality of fragments, sent through different packets. In this case, a packet corresponding to a fragment of a data message will have its field 225 completed with an indication of the position of the fragment in the data message.

The other fields are mere service fields: for example, field 223 (TTL) determines the time after which the packet may be destructed.

Figure 4b shows a second exemplary IP header (IPv6) IPH may comprise the following fields:

α

- a destination IP address 302;
- a source IP address 301;
- a version identifier 307 of a protocol being in this case the internet protocol version 6 (IPv6);
- a next header 306 designating a fragmentation header;
- other fields which do not identify a packet.

A fragmentation header IPFH is insert with the IP header IPH in order to identify the packet. The field next header 306 provides a link with the fragmentation header IPFH having the following fields:

- a next header 305 to designate another IPv6 header (if any);
- an IPv6 fragment identifier 304;
- a zone 303 containing fragmentation flags, and fragment offsets.

In this specification, "packet header" refers to information attached to a packet, and indicating e.g. the source, the destination, and other service information for versions IPv4 and IPv6.

The identification field (226 in figure 4a or 304 in figure 4b) is adapted to provide a different number for each different packet having the same other fields. By way of example only, this field 226 is 16 bits wide for the IPv4 version, is thus able to provide 65 536 different numbers and thus 65 536 different packet headers. Then, the same numbers are reused. Thus, the packet is valid during a given period of time. A filtering time period may be defined according to the time for a source node to send a given number of packets. This given number of packets may depend upon the number of different packet headers a source may provide. This notion will be hereinafter useful. As for the IPv6 version, the field 304 is 32 bits wide, the filtering time period is higher than the IPv4 filtering time period.

The operation of sending a packet Ps in redundant mode will now be described with reference to figure 5.

At 500, network protocol stack 10 of node Ni receives a packet Ps from application layer 13 through management layer 11. At 502, packet Ps is encapsulated with an IP header, in which:

- field 220 comprises the address of a destination node, which is e.g. the IP address IP_10(j) of the destination node Nj in the cluster;
- field 221 comprises the address of the source node, which is e.g. the IP address IP_10(i) of the current node Ni.

Both addresses IP_10(i) and IP_10(j) may be "intra-cluster" addresses, defined within the local cluster, e.g. restricted to the portion of a full address which is sufficient to uniquely identify each node in the cluster.

In protocol stack 10, multiple data link interface 101 has data enabling to define two or more different link paths for the packet (operation 504). Such data may comprise e.g.:
- a routing table, which contains information enabling to reach IP address IP_10(j) using two different routes (or more) to Nj, going respectively through distant interfaces IP_12(j) and IP_14(j) of node Nj;
- link level decision mechanisms, which decide the way these routes pass through local interfaces IP_12(i) and IP_14(i), respectively;
- additionally, an address resolution protocol (e.g. the ARP of Ethernet) can be used to make the correspondence between the IP address of a link level interface and its link level (e.g. Ethernet) address.

At this time, packet Ps is duplicated into two copies Ps1, Ps2 (or more, if more than two links 31, 32 are being used). In fact, the copies Ps1, Ps2 of packet Ps may be elaborated within network protocol stack 10, either from the beginning (IP header encapsulation), or at the time the packet copies will need to have different encapsulation, or in between.

At 506, each copy Ps1, Ps2 of packet Ps now receives a respective link level header or link level encapsulation. Each copy of the packet is sent to a respective one of interfaces 12 and 14 of node Ni, as determined e.g. by the above mentioned address resolution protocol.

In a more detailed exemplary realization, multiple data link interface 101 in protocol stack 10 can prepare (at 511) a first packet copy Ps1, having the link level destination address LL_12(j), and can send it through e.g. interface 12, having the link level source address LL_12(i). Similarly, at 512, another packet copy Ps2 is provided with a link level header

containing the link level destination address LL_14(j), and can be sent through e.g. interface 14, having the link level source address LL_14(i).

On the reception side, several copies of a packet, now denoted generically as Pa should be received from the network in node Nj. The first arriving copy is denoted Pa1; the other copy or copies are denoted Pa2, and also termed "redundant" packet(s), to reflect the fact that they bring no new information.

As shown in figure 6, one copy Pa1 should arrive through e.g. Link level interface 12-j, which, at 601, will de-encapsulate the packet, thereby removing the link level header (and address), and pass it to protocol stack 10(j) at 610. One additional copy Pa2 should also arrive through Link level interface 14-j which will de-encapsulate the packet at 602, thereby removing the link level header (and address), and pass it also to protocol stack 10(j) at 610.

Thus, protocol stack 610 normally receives two identical copies of the IP packet Pa, within the flow of other packets. This invention will enable (i) discriminating between a first incoming packet Pa1 and one or more redundant following packets Pa2, and (ii) filtering the packet data. The filtering will depend upon the fact a message is fragmented between several packets or not, if such a fragmentation is authorized. Since the ultimate purpose is in most cases filtering, the word "filtering", as used here, may encompass both discriminating and filtering. It should however be kept in mind that "discriminating" is the basic function.

It should now be recalled that, amongst various transport internet protocols, the messages may use the Transmission Control Protocol (TCP), when passing through function or layer 106; TCP has its own capability to suppress redundant packets but with long and unpredictable delays. The messages may also use the User Datagram Protocol (UDP), when passing through function or layer 104; the User Datagram Protocol relies on application's capability to suppress redundant packets, in the case of redundancy. This suppression of redundant packets is also long and resource consuming.

Incoming packet copies have an IP header according to figure 4. The transport protocol (TCP, UDP, or others) being used for a packet is specified in field 224 of the IP header (or, alternatively, in a separate transport protocol header).

α

This invention may be viewed as providing, at reception side, a filtering function which operates independently of the transport internet protocol being used, i.e whether e.g. TCP or UDP in the case of Internet, or another protocol. This invention is also compatible with existing transport protocols: the built-in TCP processing of redundant packets may be kept as a function; in case of UDP, the processing of redundant packets by user applications may also be kept as a function.

The applicant has also proposed another realization in PCT/IB01/01382 entitled "Filtering redundant packets in computer network equipments".

Thus, network protocol stack 10 comprises a filtering function to detect and reject redundant packets. The filtering function may be located in multi data link interface 101, or in IP layer 102, or in a distinct function module.

In accordance with an aspect of this invention, information contained in the IP headers of packets can be used for discriminating packets when they arrive to network protocol stack 10. To this effect, this information is used to build distinctive identifiers or "footprints" of the incoming packets.

As shown in figure 7, the filtering function uses a memory manager 560 having memory area (560M), and an incoming packet manager 550, comprising a set of associated (or "filtering") functions.

In a preferred realization, the memory area 560M is "reserved" statically for the filtering functions by the central processing unit (not shown) of the node; however, alternatively, the memory area 560M might also be reserved dynamically, e.g. where the time needed for memory allocation is not crucial.

The memory manager 560 may divide its memory area into portions of memory "reserved" statically at initialization time and which may be "allocated" and "released" dynamically, and individually. However, portions of memory may also be "reserved" dynamically for new routes when added in the routing table for example. These portions of memory are used to store the above mentioned distinctive identifiers or "footprints". In the example of a database

of figure 10, the term "portion of memory" corresponds to a line of the database which may be designated with a line index. A line of the database is also called a "portion of database". In the example of the table in figure 9, the term "portion of memory" corresponds to a line of the table and to parts of memory linked to this line as described hereinafter. The table of figure 9 may be sized and filled mostly at the initialization time. The term "portion of memory" may also designate other elements.

The "filtering" functions 550 may be identified for convenience as follows:

- at 551, *search()* is a function which searches for a footprint in the memory area of 560, or in a part of it;
- at 552, *write()* is a function which writes a footprint in the memory area;
- at 553, *erase()* is a function which releases a portion of the memory area;
- at 555, *forward()* is a function sending a packet to the upper layers;
- at 556, *delete()* is a function deleting or throwing away a packet;
- additionally, if it is desired to process message fragments, a *reassemble()* function at 559 gathers and reorders the fragments before they are forwarded to the upper layers, when the message is complete.

It will be noted that at least functions 551 through 553 interact with memory area 560.

In fact, the invention may be implemented by using software code, in which memory area 560 is represented by a memory manager (also denoted 560), capable of cooperating with memory hardware existing in the node, to have a reserved memory area, in which it defines

-a database, e.g. as described in figure 10, comprising a set of portions of database, each portion being designated with an index value

- a set of portions of memory, e.g. a table as described in figure 9, each portion of memory being designated with an index value . Additionally, incoming packet manager 550 contains at least some of the filtering functions 551-559, depending upon the desired implementation. Moreover, the incoming packet manager 550 is also adapted to execute the operations of a filtering method of the invention.

The filtering method will be now described with reference to figures 8. Figure 8 shows an example of flow chart of this invention.

An IP packet (Pa) reaches protocol stack 10 of its final destination node Nj.

At operation 810, the arriving IP packet comprises a source address *IP-src(Pa)* for which a value is computed. The value is a first hash value denoted *Hashv* value and computed from the source address using an hash function 557, called Hashv function, of the incoming packet manager 550. A source list defines all the source addresses IP (*IP-orig*) for which the memory manager filters the packets. The source list may be a table comprising several lines, e.g. n+1 lines and n being an integer, each being designated with an index. Index values and hashv values may be integers in the value interval [0, n]. A line index corresponds to the *hashv* value of the source address IP. Several source IP addresses can also have the same *hashv* value as hereinafter described in figure 9. Thus, in the line designated with the index matching the *hashv* value, operation 820 checks whether the source IP address (*IP-src*) of packet Pa matches the source IP address or one of the source IP addresses (*IP-orig*) stored in this line. This last checking is useful as the hash function may compute an identical value for several different addresses (denoted "collision"). For example only, an Hashv function such as the common CRC Hash function described in the following reference may be used: Knuth, D. The Art of Computer Programming, Volume 2 : Semi-numerical Methods, Chapter 5. Addison Wesley, 1981.

In an example, the addresses (including the "intra-cluster" addresses) of all the nodes in the cluster are in this source list. If desired, the list may exclude the local node. The list may also be restricted to those of the nodes in the cluster which are currently in operation.

If the node IP address *(IP-src)* is not stored in the line having the appropriated index, no filtering is done for the IP packet at operation 830; for example, the packet may be subject to a "normal processing", through conventional IP functions 102. Otherwise, the flow-chart continues at operation 840.

The filtering per se begins at 840.

At 840, a value X is computed for the incoming packet. As described hereinafter, this value comprises a union of data or fields concerning the packet (see the structure of *cgtp-pkt-footprint-t* in Exhibit A which is an example of data structure used to represent a packet

α

identifier X). Thus, one of these field represents the incoming packet link named *itf* field. A first incoming packet Pa1 and its redundant packet(s) Pa2 shall have the same value of X, except for the *itf* field. Although it is generally qualified as a distinctive packet identifier, this value X is called a footprint or an identifier hereinafter, for simplification.

Although the identifier X may be used for a research in the memory area 560, a hash value is computed with a hash function called *Hashp* using the identifier X at operation 850. This hash value is denoted *hashp* value. This hash function may compute *hashp* using all of the bits of a packet footprint X. For example only, the *Hashp* function may be a function of the minimal perfect hashing developed by Bob Jenkins.

To detect duplicated packets, an "history" of the incoming packets already received has to be continuously maintained. The memory area 560 comprises a database organized in N+1 lines and M+1 columns, N and M being integers. Each line is designated with a line index. Index values and *hashp* values may be integers in the value interval [0, N]. The intersection of a line and a column is denoted a cell C : a line is composed of M+1 cells. A line index corresponds to the *hashp* value of the identifier X of an incoming packet. In a given line, each cell may comprise a footprint X of a packet having the *hashp* value. As several incoming packets may have the same *hashp* value, several cells (and columns respectively) are forecast for a line (and lines respectively). Cells (columns respectively) are thus used in case of hash collision if the hash function does not avoid entirely collisions.

At operation 860, the identifier X of the incoming packet Pa is searched in the cells of the line designated with an index corresponding to the *hashp* value. If this identifier X is comprised in one cell C of the line, flow-chart continues at operation 870. Otherwise, it continues at operation 880.

In the database, the identifier X is recorded with its arrival time, this arrival time being the current time at the time it is recorded. A time period indicates the validity period for a recorded identifier X. The “age” of the identifier X is computed comparing the current time and its stored arrival time. If this age is greater than the time period, then the recorded identifier X of its corresponding incoming packet is considered to be “too old”, it is no more valid.

α

The operation 880 comprises to determine if a cell remains free in the line having the index corresponding to the *hashp* value. If no cell remains free, a cell is choosen in the line, this cell having the oldest identifier X in the line. This oldest identifier is deleted at operation 890. The identifier X of the incoming packet is then recorded in this cell with its arrival time at operation 897.

At operation 870, as the identifier X has been found in a cell of the line, it is checked if the recorded identifier X is not too old. If it is, the new identifier X is recorded with its current arrival time at operation 897 in the same cell of the line. If it is not too old, the new incoming packet is considered to be a redundant packet so the cell in the line may be liberated in the database for other incoming packets at operation 895. Thus, any redudant packets which arrive during the time period of the source node make room in a line.

After operation 897, the incoming packet is not redundant and is passed to the protocol stack.

The arrival time may be understood as the current time at which an operation is done for the incoming packet, for example the current time at which the X identifier of the incoming packet is recorded (qualified as the stored arrival time) or the current time at which the comparison between the age of the stored X identifier and the time period is done (qualified as the current arrival time).

The use of *hashp* index means as few comparisons as possible for the search in the database.

An example of the footprint X computation is hereinafter described refering to exhibit A. IPv4 entry, IPv6 entry and free entry of packets are all defined in lines 1 to 3. The source address of the incoming packet is mapped as an IPv6 source address structure (lines 4 to 6). Computation of the footprint X begins line 7. The source address of the incoming packet (field 221 or 301 of figures 4a et 4b) is added to a first union of different fields of the incoming packet header (for the IPv4 version) or a second union of different fields of the incoming packet header (for the IPv6 version). The first union comprises fields 224 (line 13), 225 (line 14), 222 (line 15), 226 (line 16) of figure 4a and the second union comprises fields 303 (line 22) and 304 (line 23) of figure 4b.

A time period may differ for each source node and may be adapted (or updated) dynamically according to the input packet rate of each source node. At start time, this period called ip_cgtp_filter_period is only an initial period associated to each source node recorded in the filter. Then, if one source node emits packets in a faster way than other source nodes, or if faster networks are used for some source nodes, the period per source node can be lowered dynamically. The incoming packet manager may customize a time period for packets that have the same source address.

The database size may be defined by the number of lines (ip_cgtp_filter_pkt_lines) and the number of columns (ip_cgtp_filter_pkt_collisions). For performance reasons of the IP packet hash function, the number of lines may be a power of two. For example,

ip_cgtp_filter_pkt_lines = 16384

and

ip_cgtp_filter_pkt_collisions = 3

will allow up to (16384 * (3 + 1)) =65536 IP packets recorded.

The way to compute footprint X is chosen, in combination with the internal structure of memory area 560 to reduce the risk of the two packets Pa not being redundant of each other to have the same footprint X.

It must be understood that the database comprises portions of memory, allocated by memory manager 560 within the memory area reserved to it.

The source node table (denoted filter host table) of a receiving node is illustrated on figure 9. Thus, this table T1 comprises all the source IP addresses of nodes whose emitted packets have to be filtered by the receiving node. The table comprises n+1 lines L0 to Ln each designated with a different integer index I1 comprised in the value interval [0,n]. This table comprises a first column C1-1 for first *IP-orig* addresses of source nodes for which filtering is required. A zone defines a zone memory comprising node information data such as an IP-orig address in the filter host table of figure 9 and the input packet rate of the source node corresponding to this IP-orig address. For example, the IP-orig = x is in the zone (L0, C1-1) meaning the corresponding *hashv* value is 0 and the IP-orig = y is in the zone (Ln, C1-1) meaning the *hashv* value is n. The table comprises a second column C2-1 in which, for each

line, a first part of memory is designated for the same line index. This part of memory may comprise an IP-orig address having the same *Hashv* value as the IP-orig address in column C1-1. For example, the zone (Ln, C2-1) designates a part of memory comprising a first and second zone (Ln, C1-2) and (Ln,C2-2) for an IP address having the same hashv value as the line index n. The first zone (Ln, C1-2) comprises the IP-orig = z address, its *hashv* value being n. The second zone (Ln, C2-2) is adapted to designate, for the same index n, a second part of memory (also called the next part of memory). This second part of memory may also comprise a first and second zone similar to the first part of memory. The *hashv* value is computed with a hash function as seen using the source IP address and corresponds to an index in the table. The advantage of such *hashv* value is a faster search in the table to retrieve the source IP address.

The database of a receiving node is illustrated in figure 10. Thus, this database DB comprises the received footprints of incoming packets of source nodes of table T1. The database is composed of N+1 lines Lpj and M+1 columns Ci. Each line is designated with its line index I2 whose value is in the value interval [0, N]. Each *hashp* value of an incoming packet may correspond to one of these different indexes. These indexes enable faster search in the database. The line having its index=1 in the database is now described. Each cell of this line is adapted to comprise the footprint and the arrival time of an incoming packet having its *hashp* value=1. Cell (Lp1, C1) comprises the identifier X1 and its arrival time, cell (Lp1, C2) comprises the identifier X2 and its arrival time. In case of *hashp* value collision with incoming packets, cells (Lp1, C3) and (Lp1, C4) are disposable to receive identifiers different from the recorded footprint X1 and X2. If no such footprint has been recorded yet, the footprint and its arrival time are recorded in the other columns of the same line.

The invention enables a single database to handle redundancy of all packets of all source nodes requiring filtering. Moreover, the use of hash values corresponding to indexes in the table and in the database improves the search speed for source address and footprint.

However, this invention is not limited to the hereinabove described features.

Other version of packets may be used and adapted to be handled as packets to be filtered. Other hash function may also be used.

∝

This invention also covers the software code for performing the method, especially when made available on any appropriate computer-readable medium. The expression "computer-readable medium" includes a storage medium such as magnetic or optic, as well as a transmission medium such as a digital or analog signal. The software code basically includes separately or together, the codes defining the memory manager 560, the packet manager 550, and the codes for implementing at least partially the flow-charts of figures 5, 6 and 8.

Exhibit A

cgtp_pkt_footprint_t

```
#define CGTP_ADDRESS_NONE 0 /* a free entry*/
#define CGTP_ADDRESS_IPV4 4 /* a used IPv4 entry*/
#define CGTP_ADDRESS_IPV6 6 /* a used IPv6 entry*/
typedef struct cgtp_addr_t { uint_t ipv; /* One of above CGTP-addresses*/
                        in6_addr_t addr; /* IPv6 or IPv4 mapped in IPv6*/
                    } cgtp_addr_t;

/* CGTP IP packet footprint */
typedef struct cgtp_pkt_footprint_t {
cgtp_addr_t addr; /* source address of incoming packet or free entry*/
        union {
                union {
                        struct {
                                uint8_t itf; /* incoming packet link identifier*/
                                uint8_t ip_p;/*IPv4 protocol field*/
                                uint16_t ip_frag; /*IPv4 fragmentation field*/
                                uint16_t ip_crc;/* IPv4 header CRC field*/
                                uint16_t ip_id;/*IPv4 identification field*/
                        } s4;
                } v4;
                union {
                        struct {
                                uint8_t itf ; /*incoming packet link identifier*/
                                uint16_t ip6_offlg; /*IPv6 fragmentation offset*/
                                uint32_t ip6f_id;/*IPv6 fragment identifier*/
                        } s6;
                } v6;
        } un;
} cgtp_pkt_footprint_t;
```

Claims

1. The software code, comprising code for defining:

- a memory manager (560), having a reserved memory area, in which it defines

    * a set of portions of memory comprising at least a source address, and

    * a database comprising a set of portions of database, each portion being designated with a first index value,

- an incoming packet manager (550), responsive to an incoming packet for:

    * searching the source address of the incoming packet in the set of portions of memory and

    if said source address is found:

    * determining an identifier (X) from data contained in the incoming packet and a first value from said identifier,

    * searching the identifier in a given portion of database designated with a first index value corresponding to the first value :

        * if the identifier is not found in said given portion of database, storing the identifier (X) in said given portion of database,

        * else, if a time condition is met, removing the identifier from said given portion of database, the incoming packet being a redundant packet.

2. The software code of claim 1, wherein the incoming packet manager (550) is also adapted to store the identifier in said given portion of database if the identifier is found in said given portion of database and if the time condition is not met.

3. The software code of claim 1 or claim 2, wherein the incoming packet manager (550) is adapted to store the identifier in the given portion of database and its arrival time.

4. The software code of claim 3, wherein, for an identifier (X) being found in the database, the incoming packet manager (550) is capable of comparing the current time with the stored arrival time in order to detect the age of the found identifier and is capable of comparing the age of the found identifier and a given time period in order to determine if the time condition is met.

α

5. The software code of claim 4, wherein the incoming packet manager (550) is capable of
- detecting that the time condition is met when the age of the found identifier is greater than the given time period, the found identifier being an old found identifier,
- removing the old identifier and its stored arrival time and
- replacing them by the identifier of the incoming packet and the current arrival time.

6. The software code as claimed in any of the preceding claims, wherein the incoming packet manager (550) is adapted to customize a given time period for packets having the same source address.

7. The software code of claim 6, wherein the incoming packet manager (550) is capable of updating a given time period being related to a source according to the rate of incoming packets from this source.

8. The software code as claimed in any of the preceding claims, wherein the incoming packet manager (550) is adapted to compute the identifier of an incoming packet from data of said incoming packet.

9. The software code as claimed in any of the preceding claims, wherein the memory manager (560) comprises a first hash function for determining the first value from the identifier of an incoming packet.

10. The software code as claimed in any of the preceding claims, wherein if the identifier is not found in the given portion of database and if said given portion is full of other identifiers, the incoming packet manager is adapted to delete the oldest identifier of said given portion in order to store the identifier.

11. The software code as claimed in any of the preceding claims, wherein each portion of memory is designated with a second index value and is adapted to comprise at least a source address for which the incoming packets are handled by the incoming packet manager.

12. The software code of claim 11, wherein the incoming packet manager is adapted to compute a second value from the source address of the incoming packet in order to search

the source address in a given portion of memory designated with a second index value corresponding to this second value.

13. The software code of claims 11 or 12, wherein a portion of memory designated with a second index value comprises

a first zone of memory comprising a first source address for which the second value corresponds to the second index value and

a second zone of memory adapted to designate a second portion of memory having the same second index value.

14. The software code of claim 13, wherein the second portion of memory comprises

a first zone of memory comprising a second source address for which the second value corresponds to the second index value and

a second zone of memory adapted to designate a third portion of memory having the same second index value.

15. The software code of claim 14, wherein the third portion of memory comprises a first zone of memory and a second zone of memory as indicated for the first and second portion of memory.

16. The software code of any one of the preceding claims, wherein the identifier comprises the source address.

17. A method of processing redundant packets comprising the steps of :

a- reserving memory area for

- a database comprising a set of portions of database, each portion being designated with a first index value,

- a set of portions of memory comprising at least a source address, and

responsive to an incoming packet :

b- searching the source address of the incoming packet in the set of portions of memory and if said source address is found:

c- determining an identifier (X) from data contained in the incoming packet and a first value from said identifier,

α

d- searching the identifier in a given portion of database designated with a first index value corresponding to this first value:

d1- if the identifier is not found in said given portion of database, storing the identifier (X) in said given portion of database,

d2- else, if a time condition is met, removing the identifier from said given portion of database, the incoming packet being a redundant packet.

18. The method of claim 17, wherein step d2- further comprises storing the identifier in said given portion of database if the identifier is found in said given portion of database and if the time condition is not met.

19. The method of claim 17 or claim 18, wherein step d1 or d2 comprises storing the identifier in the given portion of database and its arrival time.

20. The method of claim 19, wherein step d2 comprises comparing the current time with the stored arrival time in order to detect the age of the found identifier and comparing the age of the found identifier and a given time period in order to determine if the time condition is met.

21. The method of claim 20, wherein step d2 comprises

- detecting that the time condition is met when the age of the found identifier is greater than the given time period, the found identifier being an old found identifier,
- removing the old identifier and its stored arrival time and
- replacing them by the identifier of the incoming packet and the current arrival time.

22. The method of claim 21, wherein it comprises customizing a given time period for packets having the same source address.

23. The method of claim 22, wherein it comprises updating the time period being related to a source according to the rate of incoming packets from this source.

24. The method of claim 17, wherein step c- comprises computing the identifier of an incoming packet from data of said incoming packet.

25. The method of claim 24, wherein step c- further comprises using a first hash function for determining the first value from the identifier of an incoming packet.

26. The method of claim 19, wherein step d1 further comprises, if said given portion is full of other identifiers, comparing the current time with the stored arrival times of said other identifiers in order to detect their ages, to detect the oldest identifier of said given portion and to delete it in order to store the identifier and its current arrival time.

27. The method of claim 17, wherein each portion of memory of step a is designated with a second index value and comprises at least a source address for which the incoming packets are handled in steps b, c and d.

28. The method of claim 27, wherein step b- further comprises

b1- computing a second value from the source address of said incoming packet

b2- searching the source address in a given portion of memory designated with a second index value corresponding to this second value.

29. The method of any of claims 17 to 26, wherein the identifier comprises the source address.

30. A software product for performing the method as claimed in any of claims 17 to 29.

α (23 pages)

CABINET NETTER

1/8

FIG.1

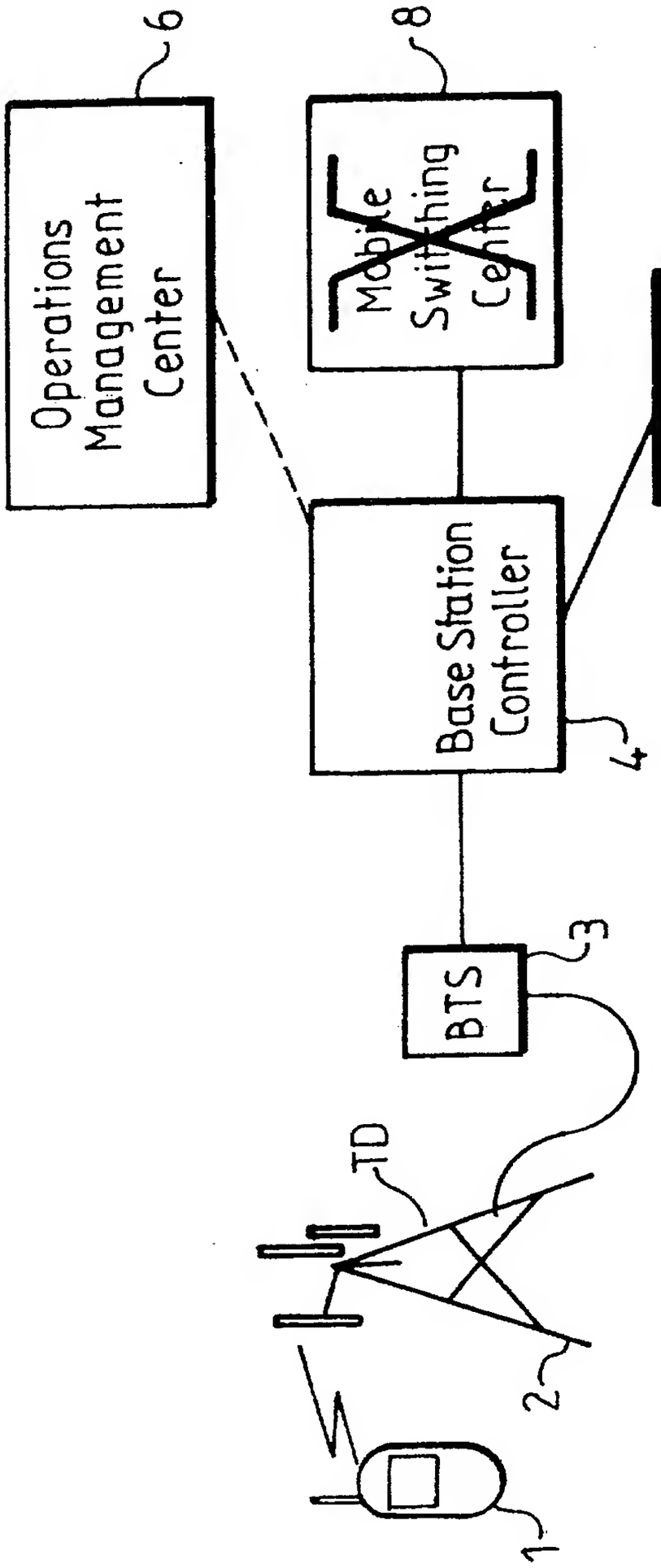
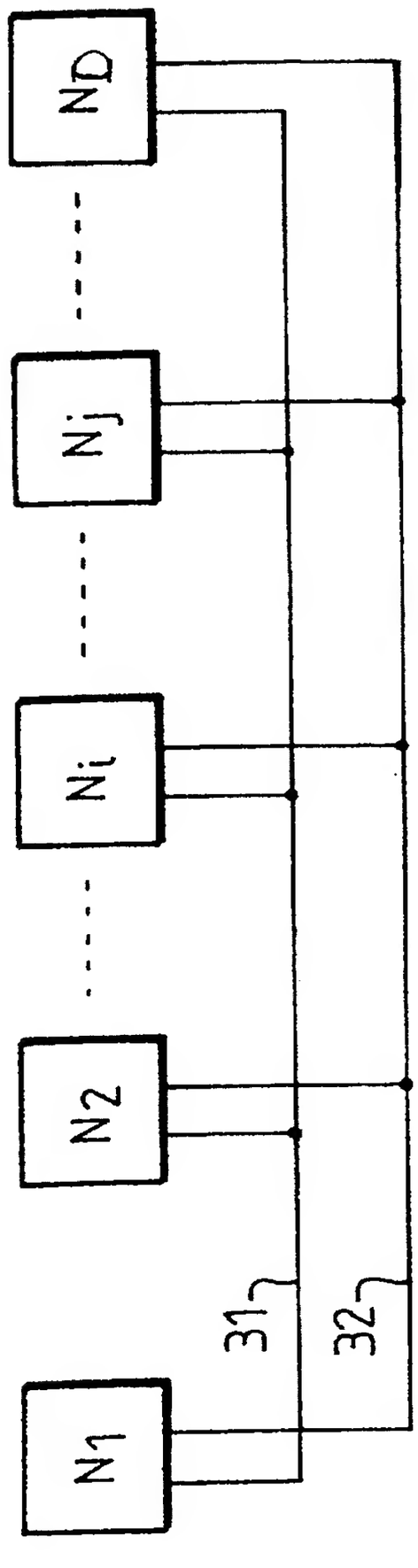


FIG.2



CABINET NETTER

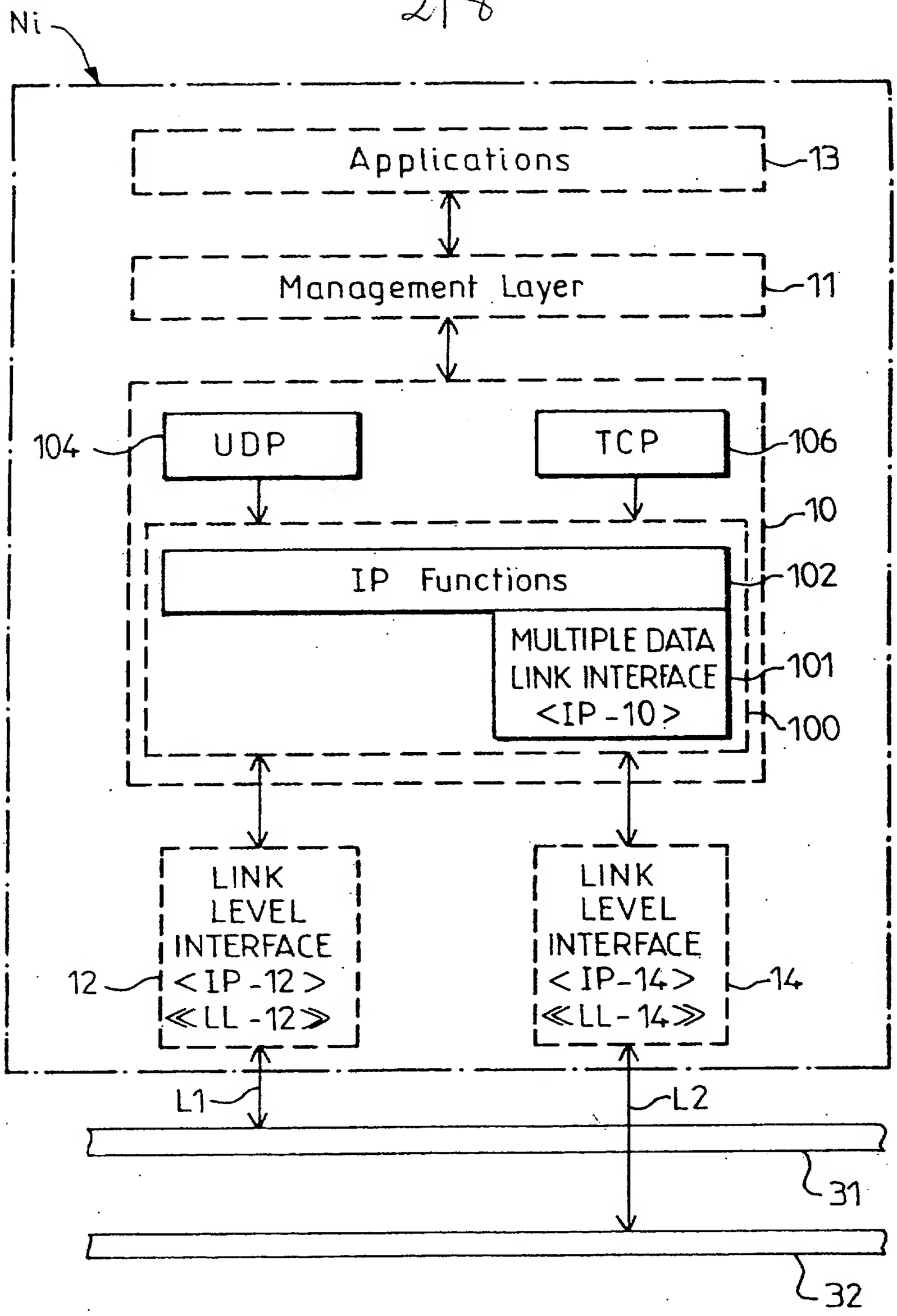


FIG. 3

CABINET NETTER

3/8

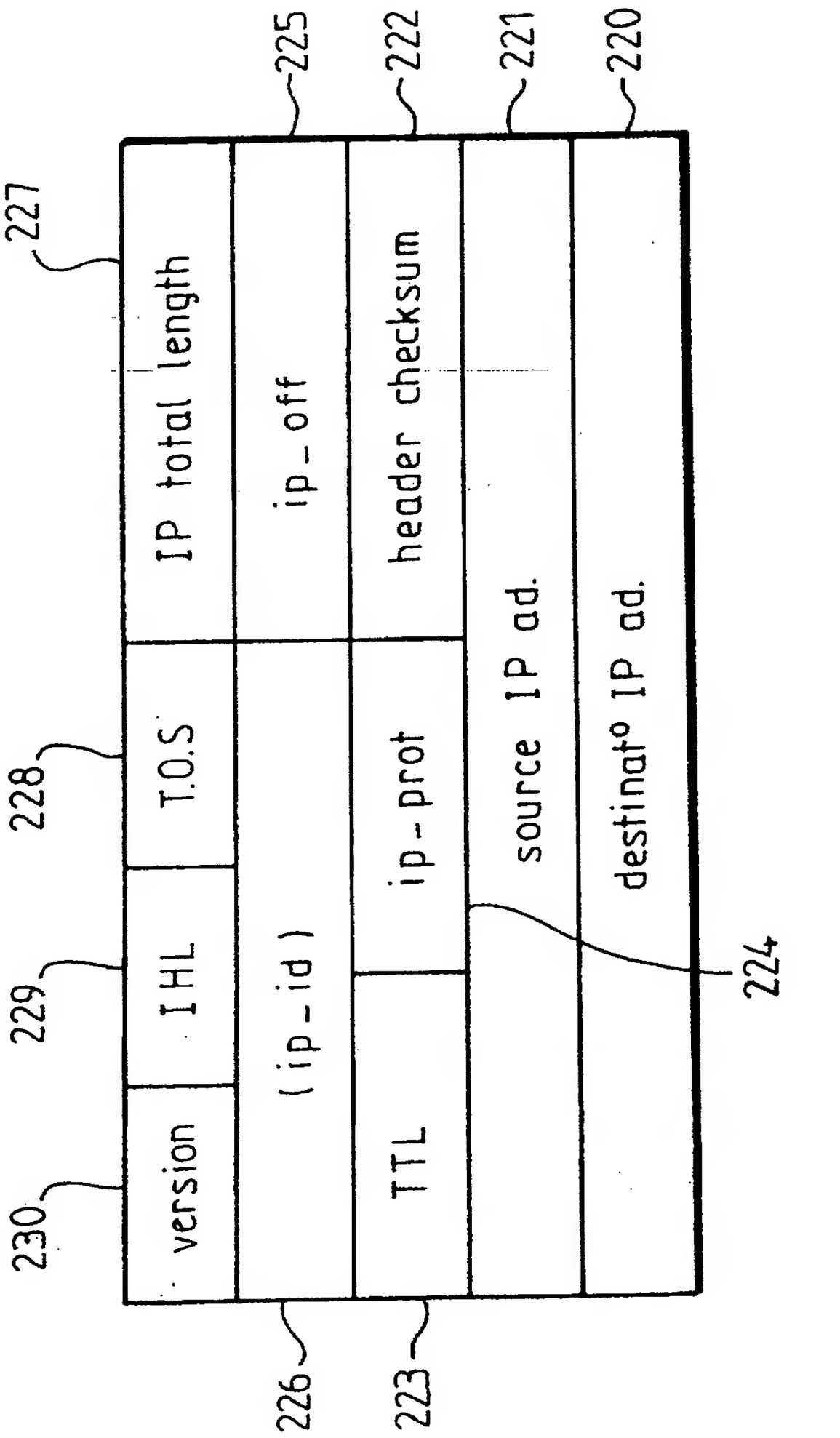


FIG.4

CABINET METTER

418

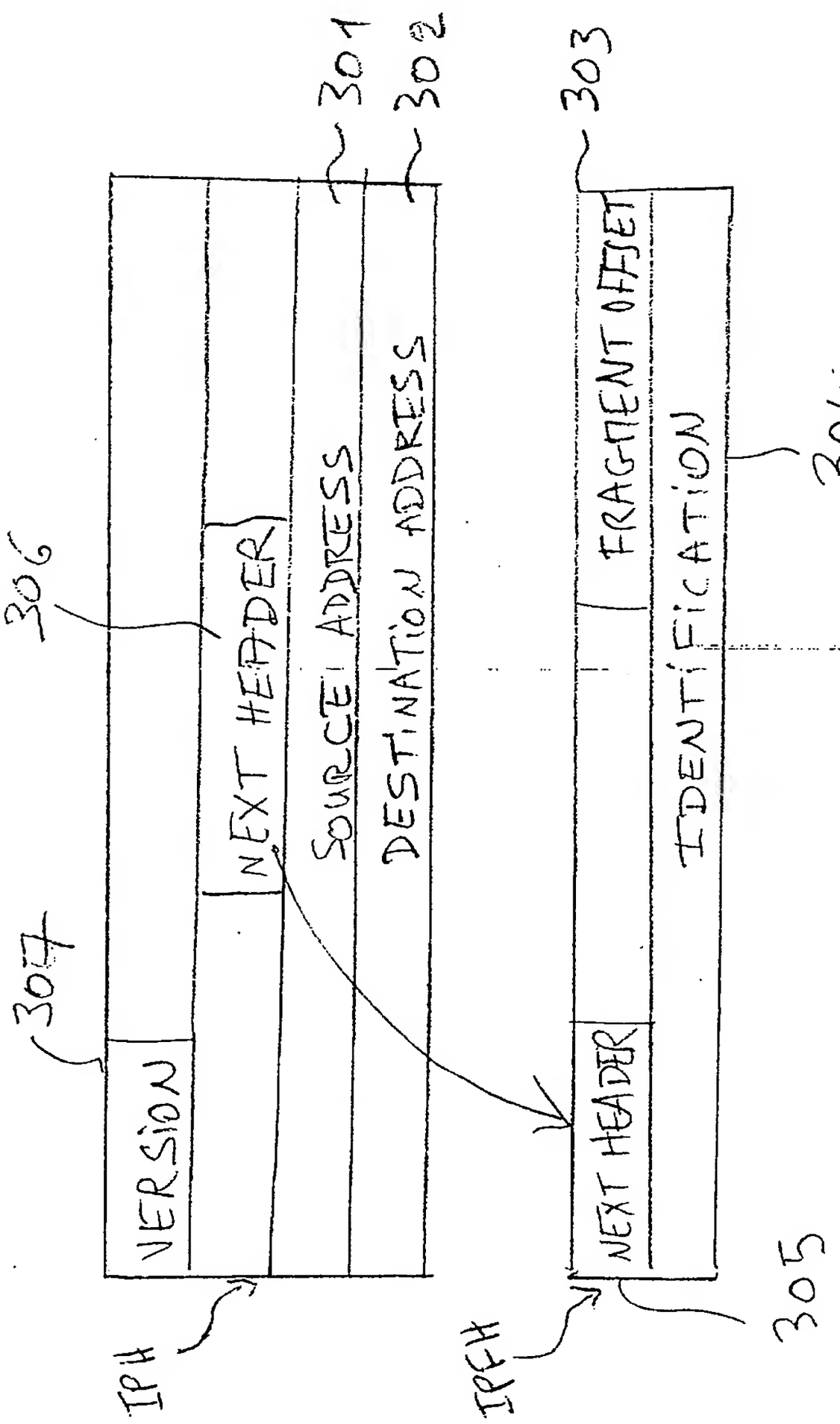


FIG 4B

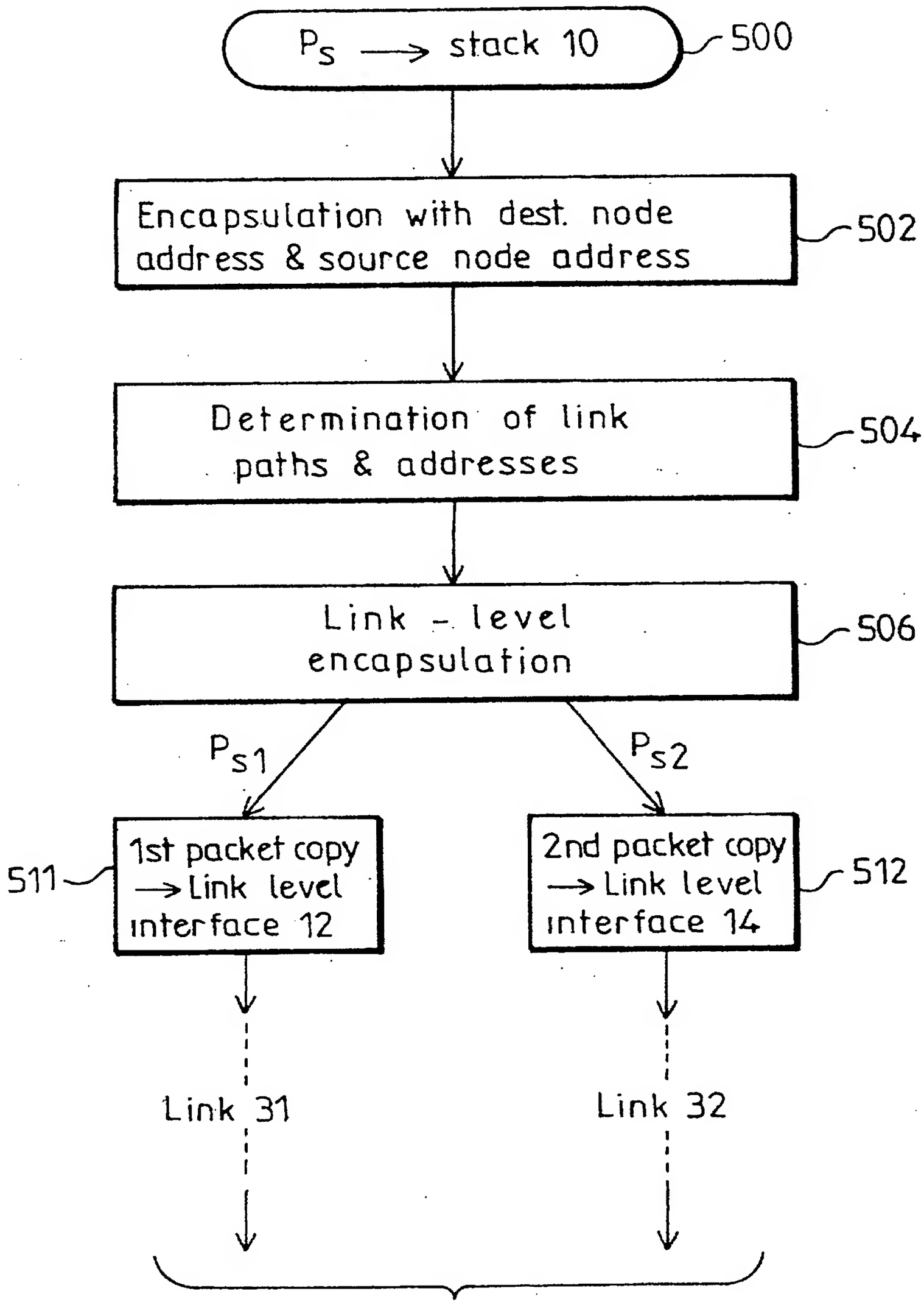


FIG.5

618

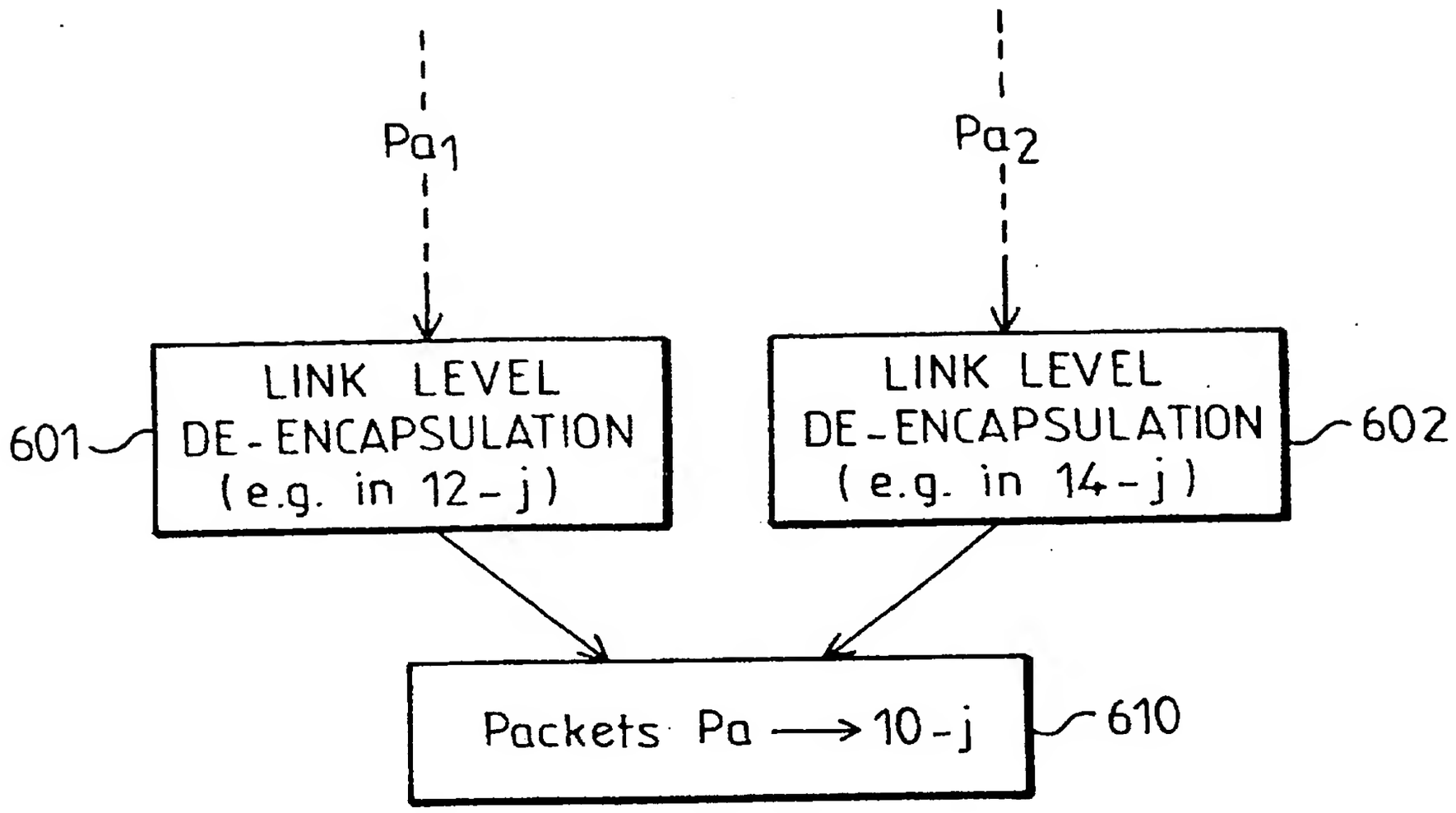


FIG.6

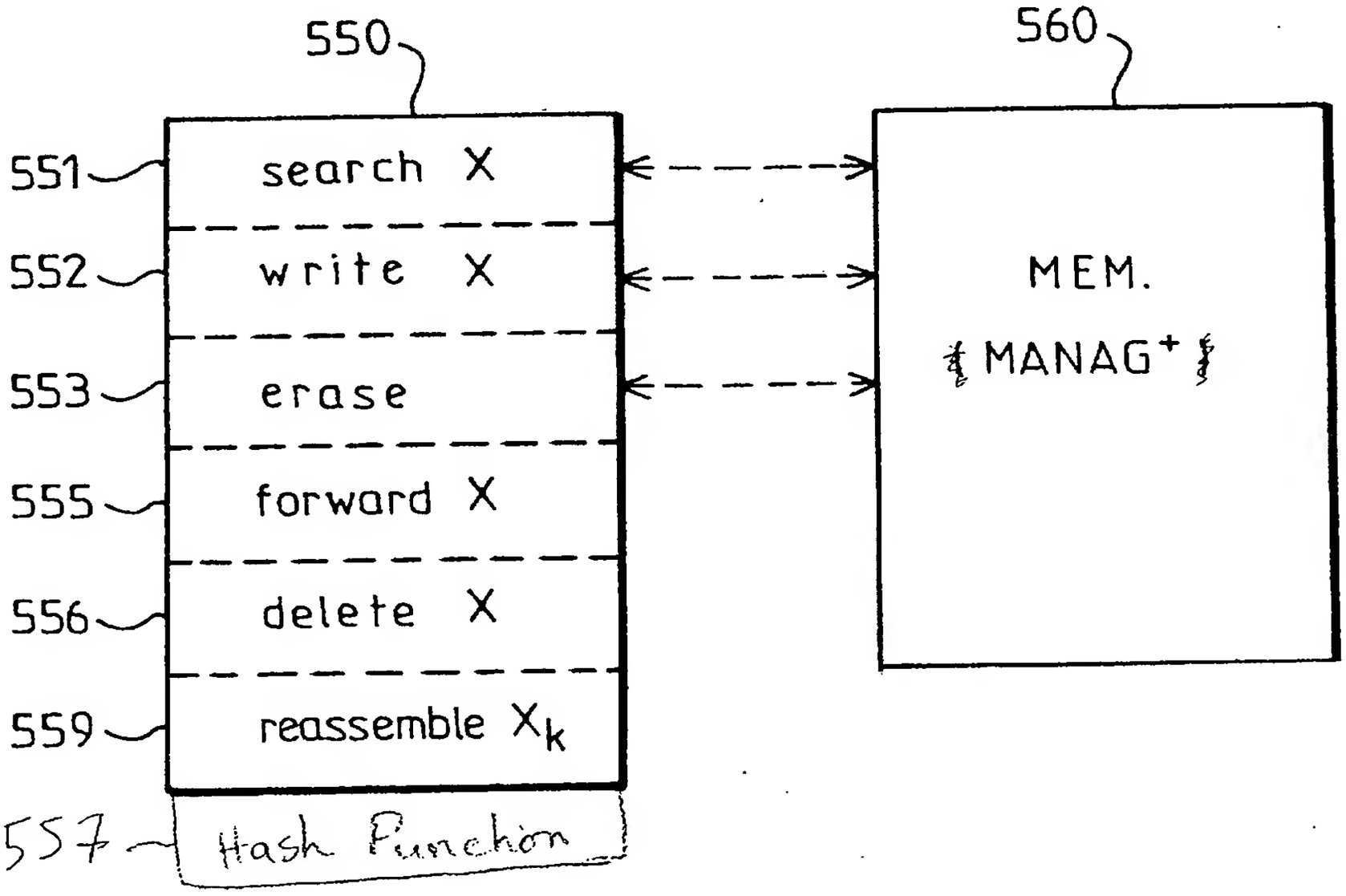
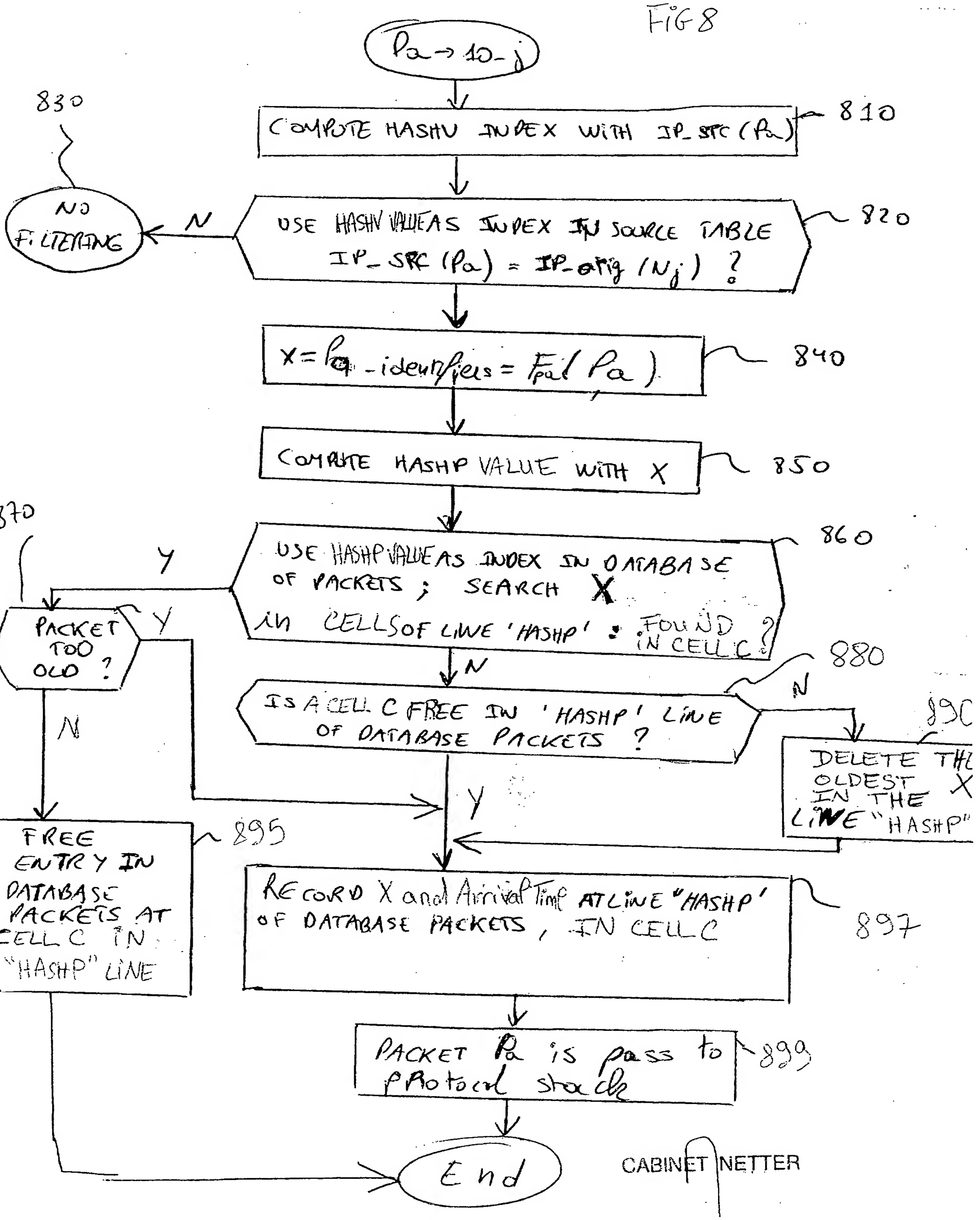


FIG.7

CABINET NETTER

FIG 8



CABINET NETTER

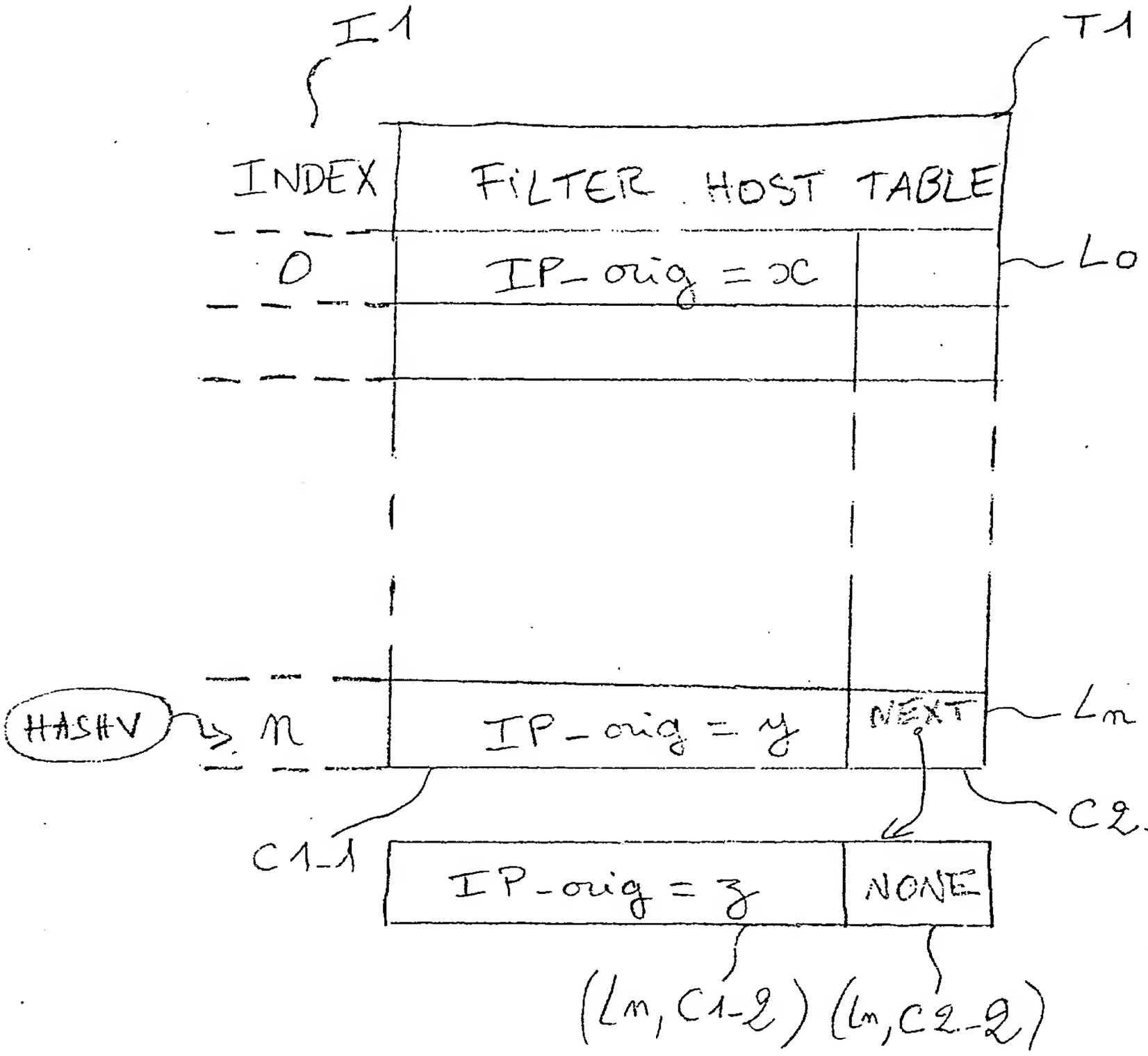


FIG 9

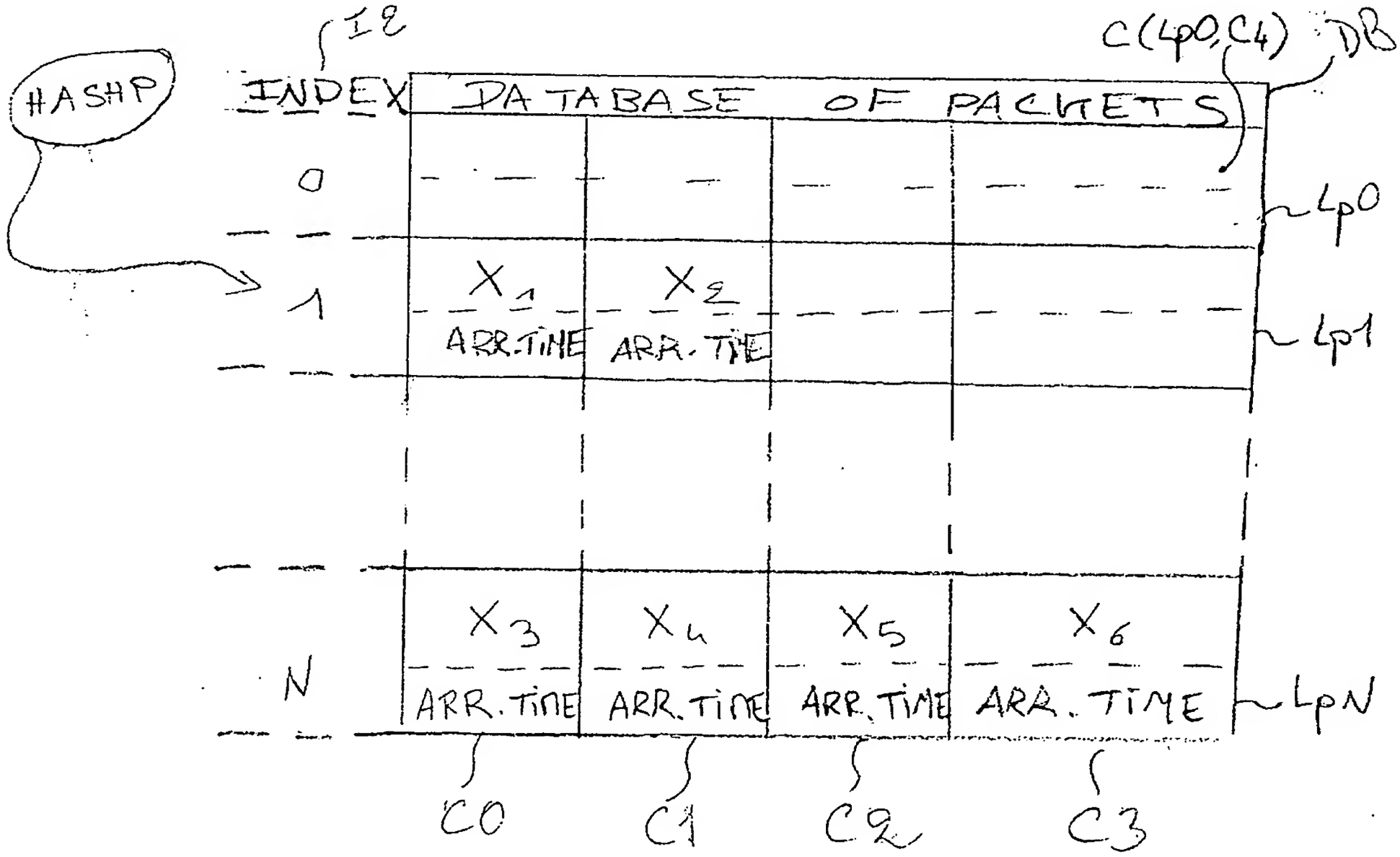


FIG 10